Tópico 07 - Entrada/Saída

Hugo Vinícius Leão e Silva

hugovlsilva@gmail.com, hugo.vinicius.16@gmail.com, hugovinicius@ifg.edu.br

Instituto Federal de Educação, Ciência e Tecnologia de Goiás
Campus Anápolis
Curso de Bacharelado em Ciência da Computação

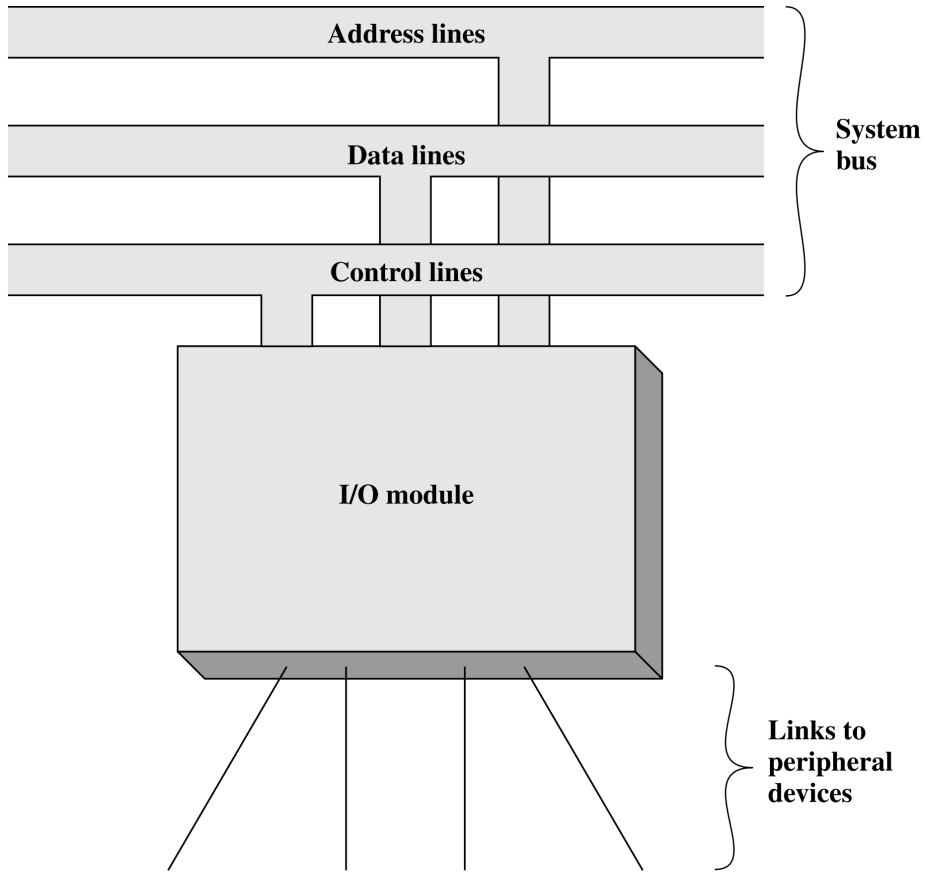
11 de Abril de 2021

Visão geral

Introdução

- Computador = CPU + RAM + E/S;
- Dispositivos de E/S se conectam a módulos/controladores de E/S;
- Controladores de E/S possuem os circuitos que permitem a comunicação entre os dispositivos de E/S e o barramento;
- Dispositivos de E/S não são conectados diretamente no barramento de sistema, pois:
  - Diversos tipos de dispositivos de E/S e modos de operação $\rightarrow$ impraticável um circuito controlador dentro da CPU;
  - *Throughput* muito menor do que da CPU ou RAM $\rightarrow$ impraticável conectá-los ao barramento de sistema;
  - *Throughput* de alguns disp. de E/S é maior do que da CPU ou RAM $\rightarrow$ ineficiente se não gerenciado apropriadamente;
  - Disp. de E/S frequentemente possuem tamanho de palavra e formatos dos dados diferentes do computador.

Introdução

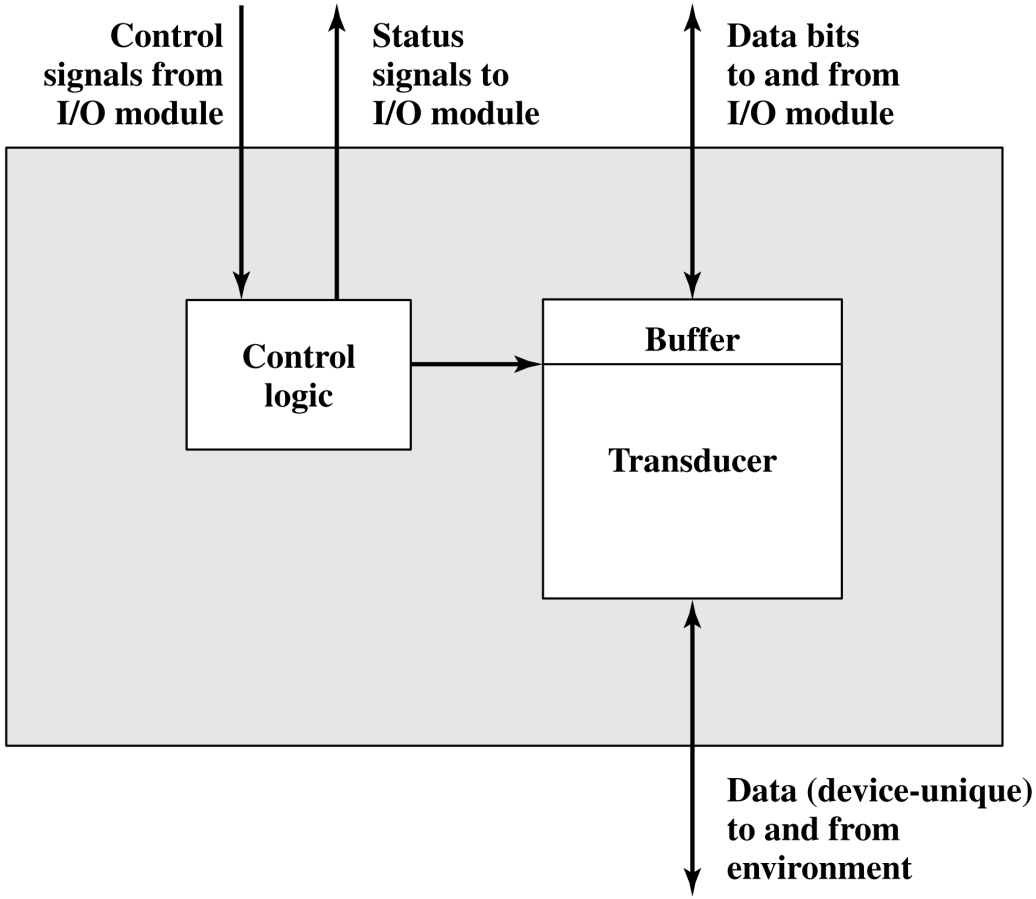


Um controlador de E/S interfaceia um ou mais dispositivos de E/S e CPU e RAM usando *links* apropriados.

Dispositivos externos

- Dispositivos de E/S (aka periféricos ou dispositivos periféricos) permitem trocar dados entre o computador e o mundo externo;
- O *link* é utilizado para trocar sinais de controle e estado (*status*) e dados entre controlador e dispositivo;
- Há basicamente três tipos de dispositivos:
  - ***Human readable*** – apropriado para comunicação com um usuário: monitores, auto-falantes, microfones etc;
  - ***Machine readable*** – apropriado para comunicação com outros dispositivos: HDs, SSDs, atuadores e sensores utilizados na automação;
  - ***Communication*** – apropriado para comunicação com dispositivos remotos (*human- or machine-readable devices*): um terminal burro ou outro computador.

Dispositivos externos



- **Controle**: `READ/INPUT`, `WRITE/OUTPUT` etc;
- **Status**: `READY/NOT-READY` ou `BUSY` etc.

Dispositivos externos

Teclado/monitor:

- Interface usuário/computador mais comum (embora devemos considerar o *touchscreen*);
- Usuário digita e, às vezes, o monitor mostra os dados inseridos;
- Unidade básica de troca: caractere. O formato mais comum é o IRA[1] (*International Reference Alphabet*), recomendação ITU-T T.50;
- Há caracteres imprimíveis e de controle (não-imprimíveis);
- Embaixo de cada tecla há um transdutor que gera a sequência de bits de acordo com a codificação;
- Às vezes, os dados no computador podem ser armazenados como IRA ou pode ser recodificado.

[1] ASCII é um subconjunto do IRA.

Controladores de E/S

- O controlador de E/S abstrai a complexidade dos dispositivos de E/S para a CPU. Na maioria das vezes, ele esconde detalhes como temporização, formatos de dados, eletromecânica etc. Faz isso como?
- **Controle e temporização**: coordenar o fluxo de dados entre recursos internos e externos. Em outras palavras, sincronizar as demandas erráticas da CPU e os dispositivos, barramento, RAM etc. Exemplo[2]:
  1. CPU pergunta ao controlador de E/S pelo status de um dispositivo;
  2. Controlador de E/S retorna o *status*;
  3. Se o dispositivo estiver pronto para transmitir (READ), a CPU faz a requisição de dados enviando um comando ao controlador;
  4. O controlador obtém uma palavra de dados do dispositivo;
  5. OS dados são transferidos à CPU.

[2]Sem considerar o acesso ao barramento!

Controladores de E/S

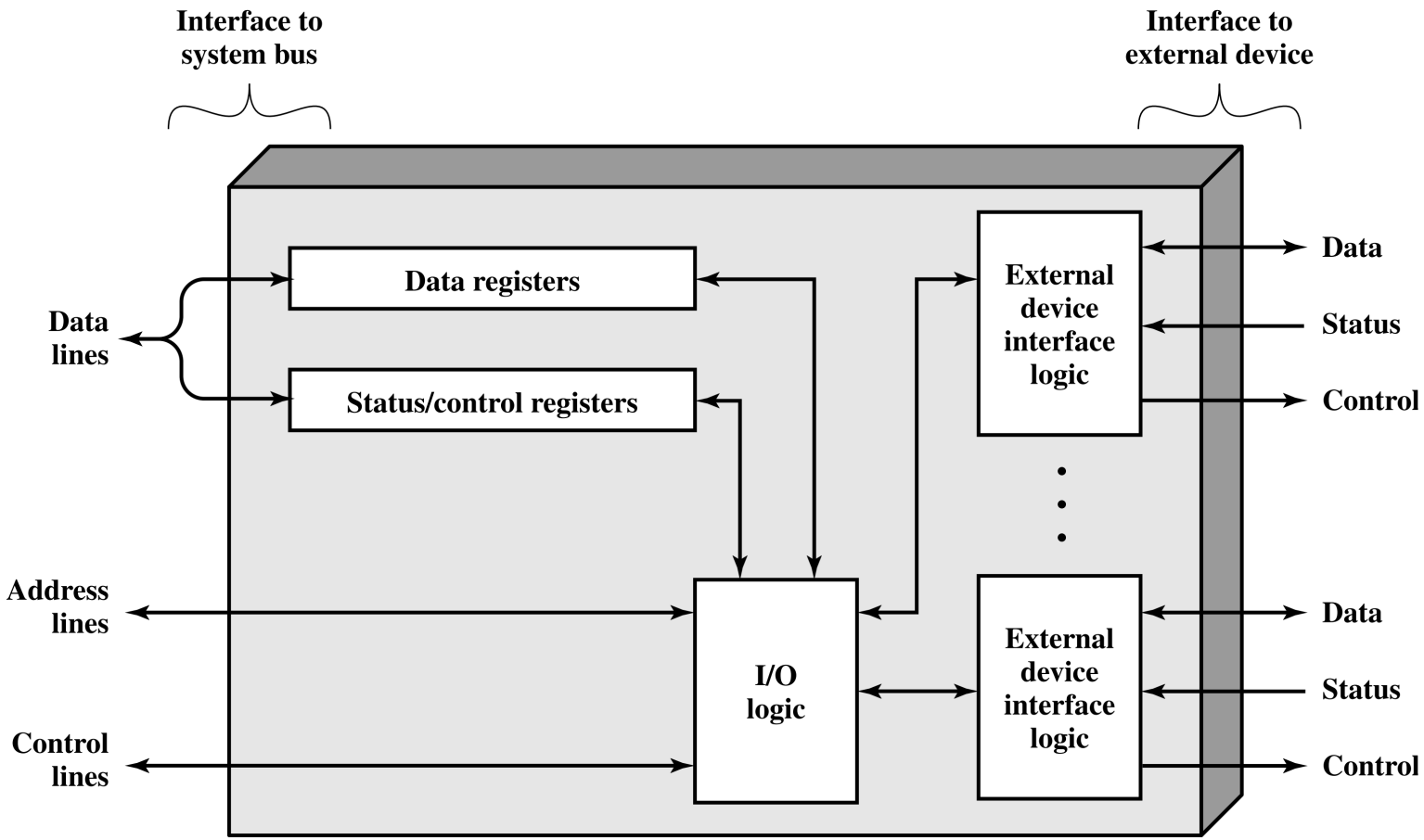
- **Comunicação com a CPU** envolve:
  1. **Decodificar comandos vindos da CPU** via linhas de controle no *bus* `READ SECTOR`, `WRITE SECTOR`, `SEEK` ou `SCAN` etc.;
  2. **Transferir dados** nas linhas de dados do *bus*;
  3. **Relato de estado** dos dispositivos: `READY`/`NOT-READY`/`BUSY` ou erros;
  4. **Reconhecimento de endereços** para identificar um endereço único de cada dispositivo controlado pela controladora de E/S.
- **Comunicação com dispositivo** para transferir dados, comandos e *statuses* entre controlador de E/S e dispositivos.

Controladores de E/S

- ***Buffering* de dados** para transferir dados entre CPU ou RAM e dispositivos de E/S e vice-versa. Geralmente: transferências CPU/RAM-controladora se dão em rajadas (*bursts*) e transferências controladora-dispositivos são lentas.
- **Detectar (e relatar) erros (à CPU)** para informar erros mecânicos ou elétricos: *paper jams* e *bad blocks* ou erros na transmissão para serem detectados e, se possível, corrigidos via ECC.

Dispositivos externos

Figura: Estrutura de uma controladora de E/S



O circuito de E/S age como um mux;

Técnicas de E/S – E/S Programada

- Na E/S Programada (*Programmed I/O* – PIO) os dados são transferidos entre memória e controlador de E/S via CPU, que executa o programa que controla diretamente a operação de E/S;
- A CPU deve esperar pela finalização de cada operação de E/S enviado – ineficiente!

1. Instruções relativas a I/O são executadas ao enviar um comando para o controlador de E/S;
2. O controlador realiza a ação e seta os bits no registrador de *status* de E/S...e só;
3. O controlador não avisa a CPU sobre nada. A CPU que tem que tomar as ações necessárias periodicamente.

Técnicas de E/S – E/S Programada

Comandos de E/S enviados pela CPU:

- **Control**: ativa um dispositivo e diz a ele o que fazer. São comandos específicos ao tipo de dispositivo. Exemplo: rebobinar fita;
- **Test**: testa varias condições de *status* associados ao controlador de E/S e dispositivos: Exemplos: dispostivo ligado e pronto? Operação de E/S completa?
- **Read**: faz o controlador de E/S ler dados de um dispositivo e colocar em um *buffer*. A leitura pela CPU pode ser feita enviar um comando para disponibilizar os dados no **bus**;
- **Write**: faz o controlador de E/S obter uma palavra de dados do *bus* e transmiti-la ao dispositivo;

- Há uma semelhança muito grande entre as instruções de acesso à memória e os comandos de E/S;
- *Reminder*: Cada módulo de E/S e cada dispositivo possui um endereço exclusivo;
- Quando CPU, RAM e controladores de E/S compartilham o barramento, pode-se utilizar **E/S mapeada em memória** (*memory-mapped I/O*) e **E/S isolada**.

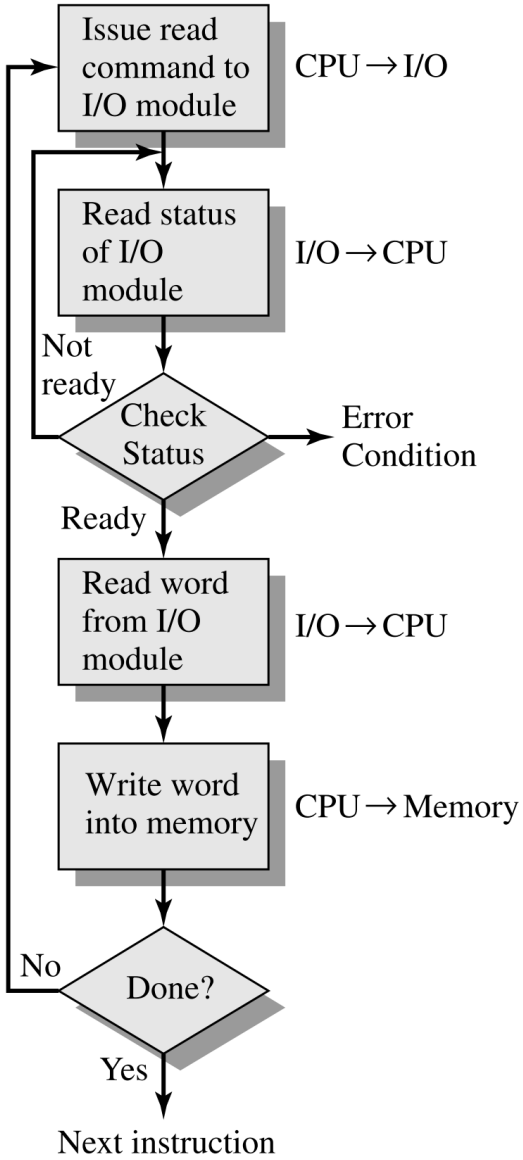
Memory-mapped I/O:

- Existe um único espaço de endereçamento para memória e E/S que pode ser alocado livremente ($2^{10} = 1024$ posições de memória ou dispositivos de E/S);
- Registradores de dados e *status* dos módulos de E/S são vistos/tratados como endereços de memória;
- As mesmas instruções para ler da/gravar na memória são utilizadas para E/S.

E/S Isolada:

- Existem dois espaços de endereçamento: um para memória e outro para E/S;
- $2^{10} = 1024$ posições de memória e $2^{10} = 1024$ dispositivos de E/S;
- Instruções diferentes para memória e para E/S.

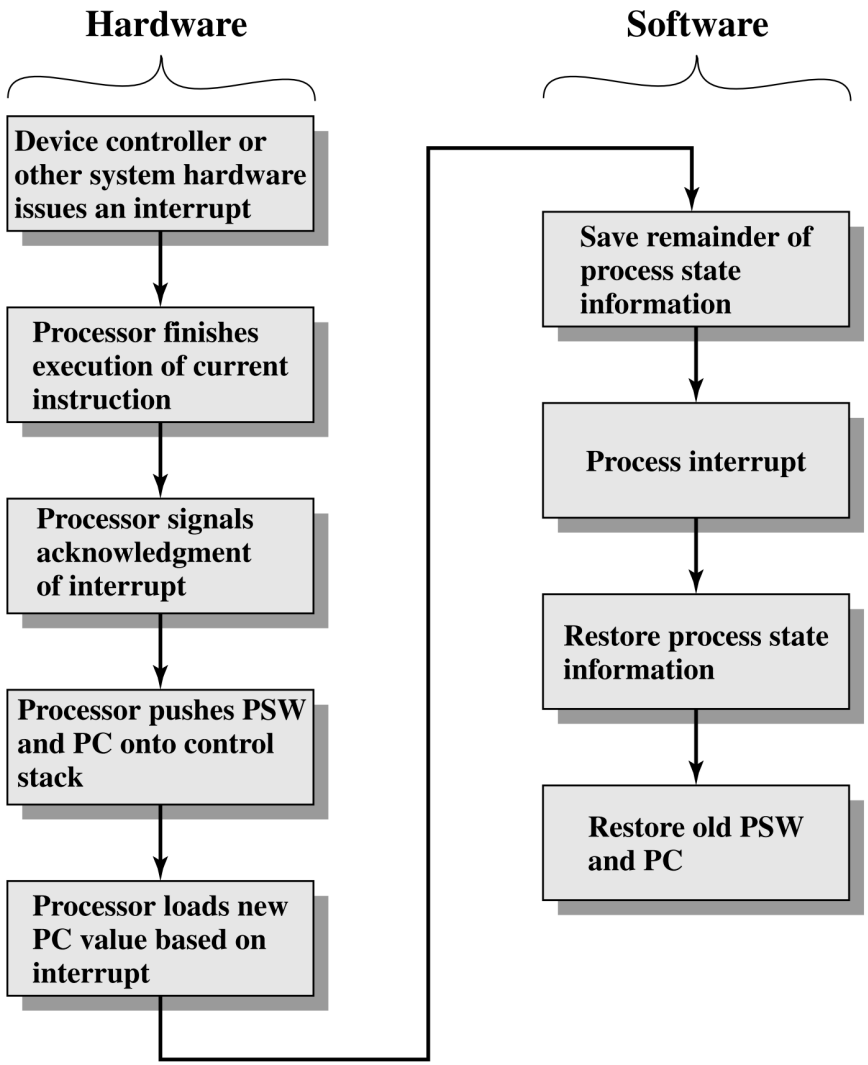
Técnicas de E/S – E/S Programada



Técnicas de E/S – E/S Controlada por Interrupção

- Já sabemos que PIO é ineficiente – desempenho do sistema é severamente degradado;
- Uma alternativa que já sabemos é E/S Controlada por Interrupção (*Interrupt-Driven I/O*);
- A CPU envia um comando de E/S para o controlador de E/S e vai fazer outra coisa;
- Quando o controlador estiver pronto para transferir dados com a CPU, envia um sinal de requisição;
- A CPU responde com um ACK (*acknowledgment*) trata a requisição e depois volta a fazer outra coisa.

Técnicas de E/S – E/S Controlada por Interrupção



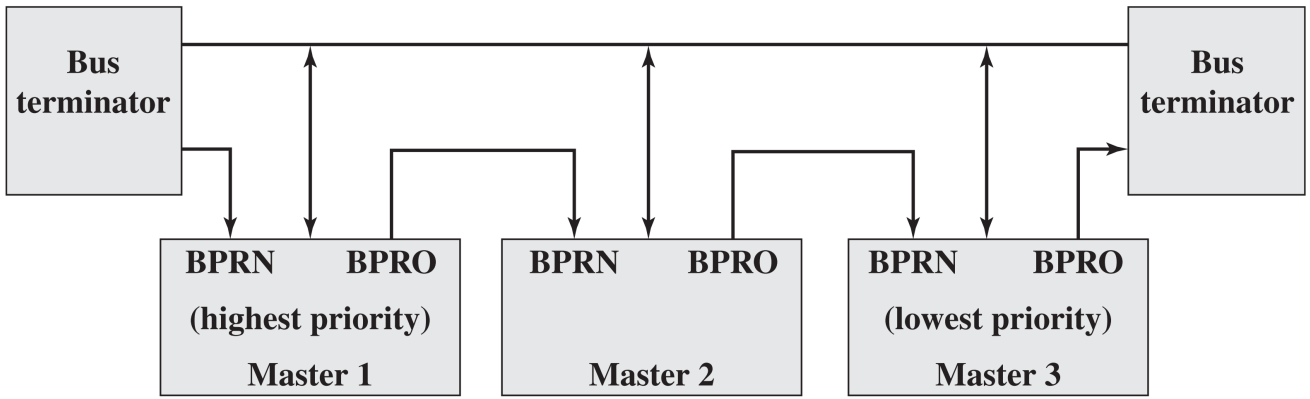
PSW – *Program Status Word*

Problemas: 1) como a CPU determina qual dos controladores de E/S enviou a interrupção? 2) E quando tem mais de uma interrupção, como decidir qual será tratada?
Soluções para o problema 1:

- **Múltiplas linhas de interrupção**: é a solução mais simples, mas pode ser impraticável por usar ocupar linhas do *bus*;
- **Verificação por software**/*Software poll*: quando uma interrupção é detectada pela CPU, ela executa um programa que pergunta para cada módulo para determinar quem enviou o sinal de interrupção. Daí ele executa o programa específico para aquele dispositivo. O problema é que esse processo leva tempo.

Soluções para o problema 1:

- *(To)* ***Daisy Chain***: na prática, é um *hardware poll*. A CPU detecta a interrupção e manda um ACK, que é recebido por todos os controladores de E/S que estão **ligados em série**:



O controlador responde com uma palavra (*vetor*) que contém o endereço do controlador ou outro identificador único. O vetor é o ponteiro para o início do programa tratador de interrupção. Chamado de **interrupção vetorada**.

Técnicas de E/S – E/S Controlada por Interrupção

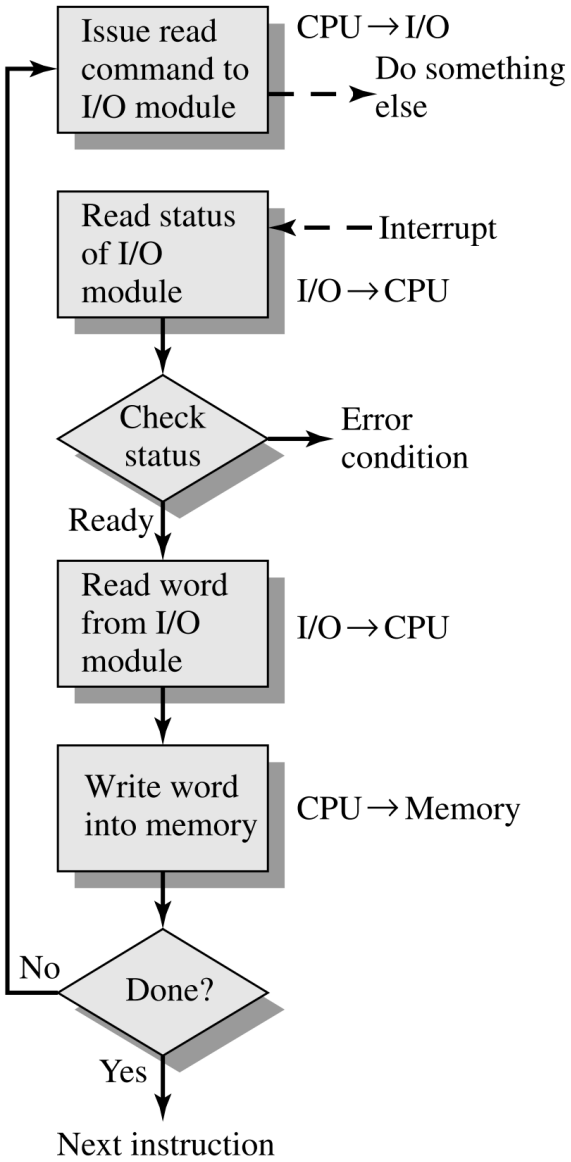
Soluções para o problema 1:

- **Arbitração de barramento**: também utiliza interrupções vetoradas. O controlador de E/S deve primeiro ter acesso ao barramento para enviar a interrupção. A cada instante de tempo, apenas um controlador pode fazer isso. A CPU responde com ACK e o controlador responde com o vetor.

Soluções para o problema 2 para tratar múltiplas interrupções:

- **Múltiplas linhas**, a CPU simplesmente considera a linha com maior prioridade;
- ***Software poll***, os módulos são verificados na ordem de prioridade;
- ***Daisy chaining***, é só cascatear os módulos na ordem de prioridade;
- **Arbitração de barramento** pode naturalmente prover um esquema de prioridade.

Técnicas de E/S – E/S Controlada por Interrupção



Técnicas de E/S – Acesso Direto à Memória (DMA)

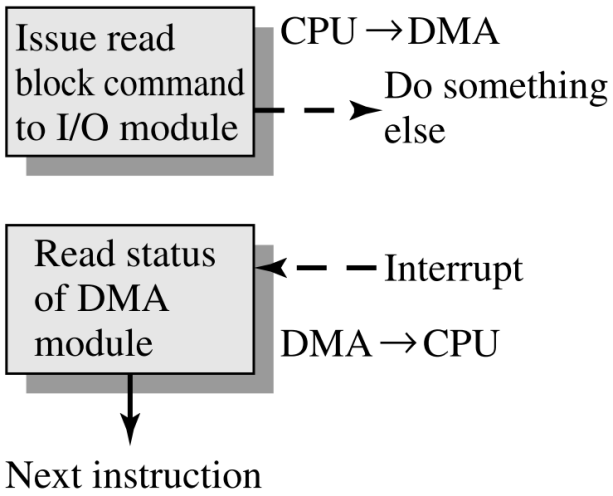
- E/S Controlada por Interrupção é mais eficiente do que o PIO, mas ainda requer a intervenção da CPU;
- Toda transferência entre RAM e dispositivo de E/S **precisa** passar pela CPU;
- Há dois problemas:
  - O *throughput* de E/S fica limitada pela velocidade que uma CPU pode testar e enviar comandos a um dispositivo;
  - A CPU fica ocupada com o gerenciamento de E/S. Cada transferência de E/S exige a execução de diversas instruções.
- Um meio mais eficiente para transferir grandes volumes de dados é o Acesso Direto à Memória (*Direct Memory Access* - DMA).

Técnicas de E/S – Acesso Direto à Memória (DMA)

- O DMA exige um controlador adicional no *system bus*;
- Esse controlador "imita" a CPU e assume o controle do sistema para transferir dados entre ele e a RAM pelo *system bus*;
- Entretanto, ele só pode fazer isso quando a CPU não está usando o *bus* ou deve forçar uma suspensão da CPU – *cycle stealing*;
- Quando a CPU deseja fazer uma operação de E/S, envia um comando para o controlador de DMA contendo:
  - Operação de leitura ou gravação?
  - Endereço do dispositivo de E/S;
  - Posição inicial na memória para ser lido ou escrito;
  - O número de palavras a serem lidas ou escritas.

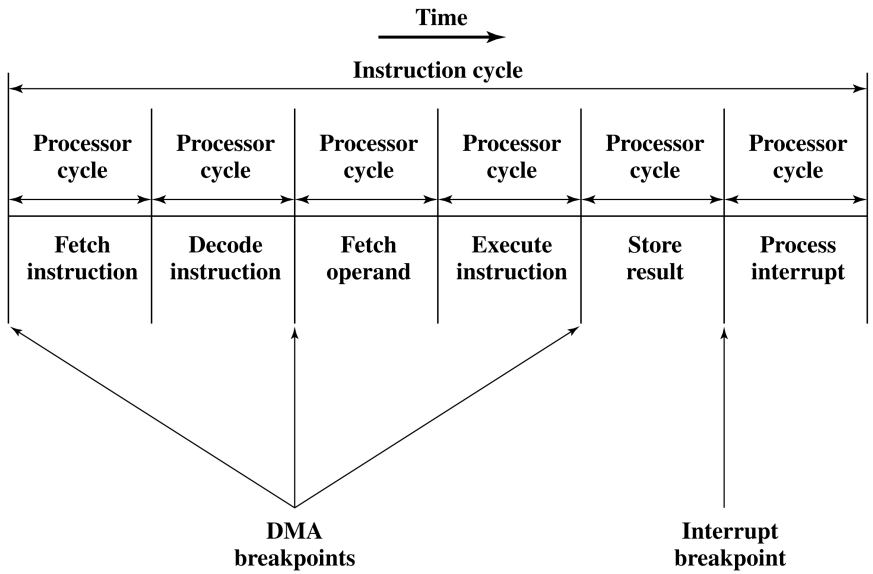
Técnicas de E/S – Acesso Direto à Memória (DMA)

- Depois disso, o processador continua com outras tarefas e o controlador DMA fica responsável pela operação de E/S;
- Ele transfere todo o bloco de dados, palavra por palavra, da/para a RAM diretamente, sem envolver a CPU;
- Quando a operação é finalizada, o módulo DMA envia um sinal de interrupção;
- A CPU só é envolvida diretamente apenas no início e no fim da operação de E/S.



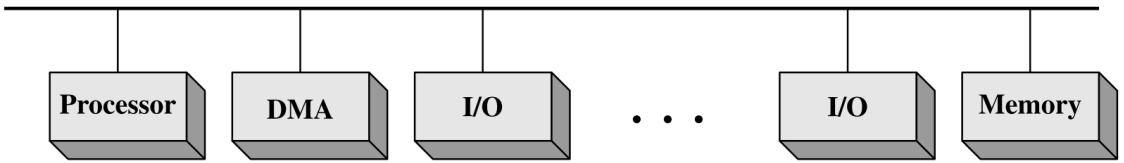
Técnicas de E/S – Acesso Direto à Memória (DMA)

- O controlador DMA apenas usa o barramento quando está livre. Somente em alguns instantes onde o ciclo de instrução pode ser suspenso:

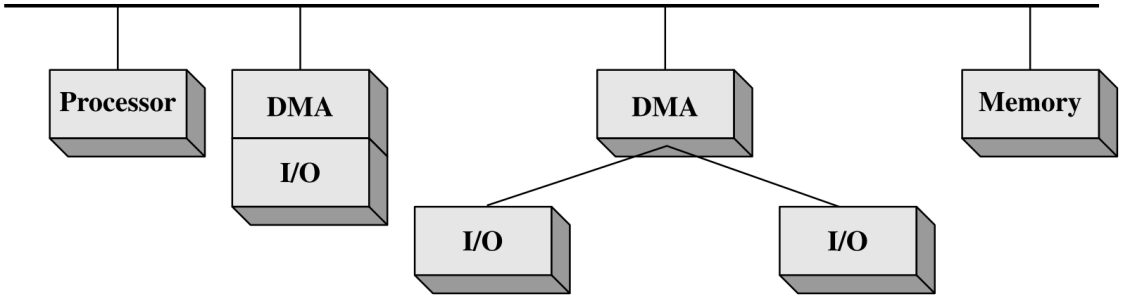


- O ciclo de instrução fica *um pouco* mais lento, mas para grandes transferências via DMA são mais eficientes do que PIO ou por interrupção;
- O controlador transfere uma palavra por vez e retorna o controle do *bus* para a CPU, que não precisa salvar o contexto do programa ou fazer outra coisa.

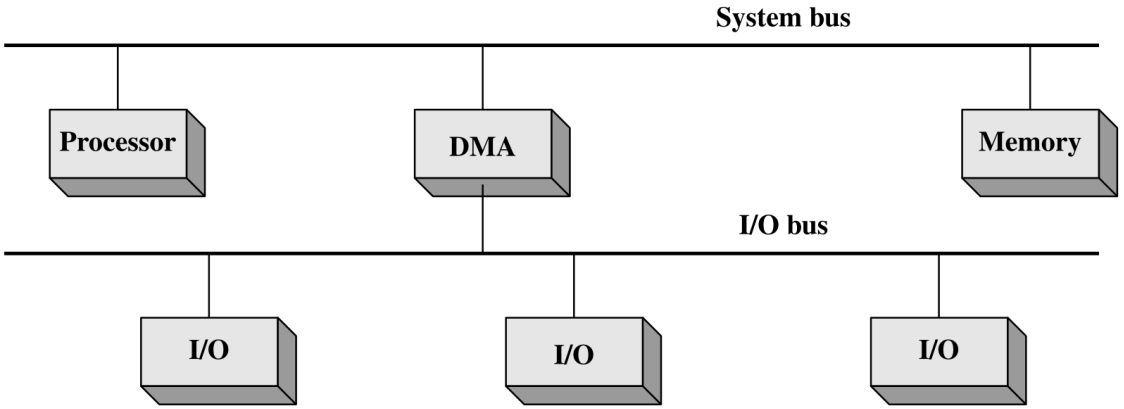
Técnicas de E/S – Acesso Direto à Memória (DMA)



(a) Single-bus, detached DMA



(b) Single-bus, integrated DMA-I/O



(c) I/O bus

Processadores e Canais de E/S

A evolução dos mecanismos/circuitos de E/S pode ser resumida da seguinte forma:

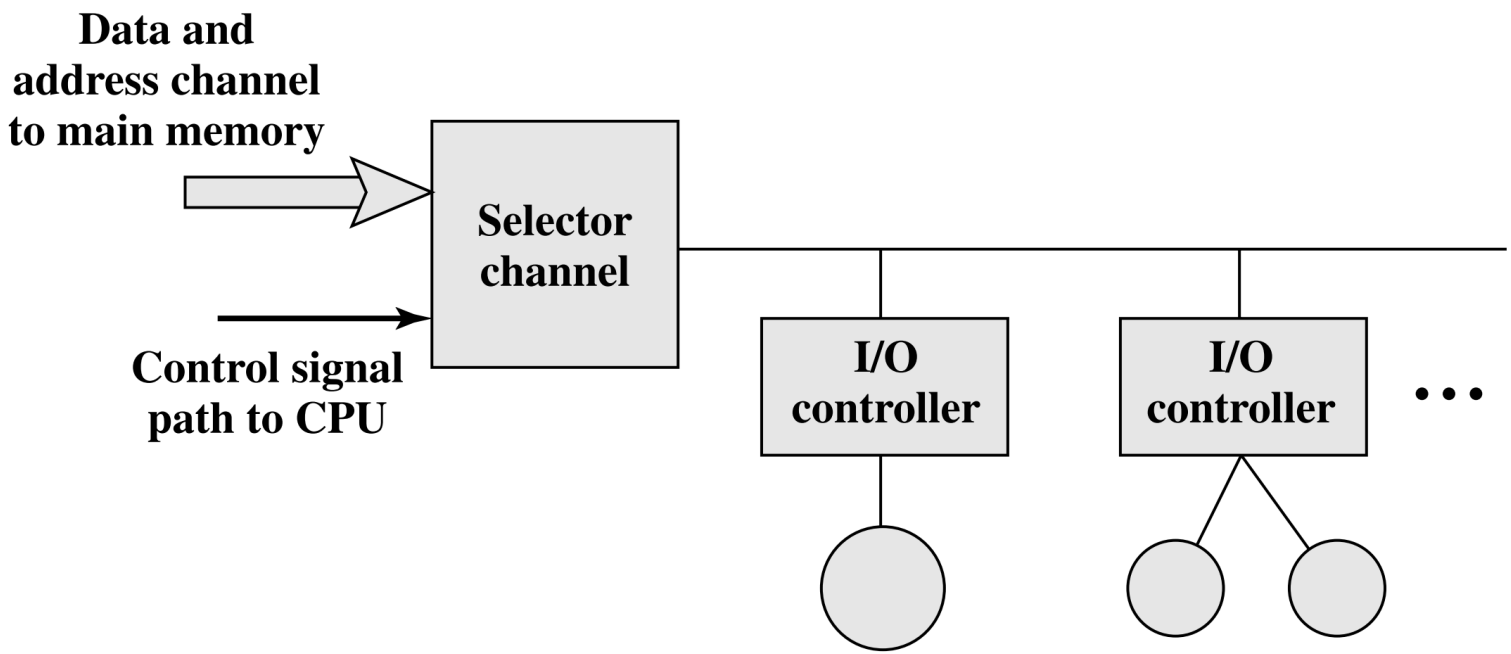
1. A CPU controla diretamente um dispositivo de E/S, como é em computadores simples controlados por CPU;
2. Um controlador de E/S é adicionado e utiliza o PIO;
3. Implementa-se a E/S Controlada por Interrupção;
4. Um controlador de DMA tem acesso direto à memória e faz operações de E/S sem envolver a CPU;
5. Ele mesmo é uma CPU – ou seja, programável – com conjunto de instrução apropriado para E/S;
6. Também tem sua própria memória e, assim, é um computador. Muitos tipos de dispositivos de E/S podem ser controlados com intervenção mínima da CPU principal.

Itens 5 e 6 são chamados intercambiavelmente de Canais de E/S ou Processadores de E/S.

Processadores e Canais de E/S

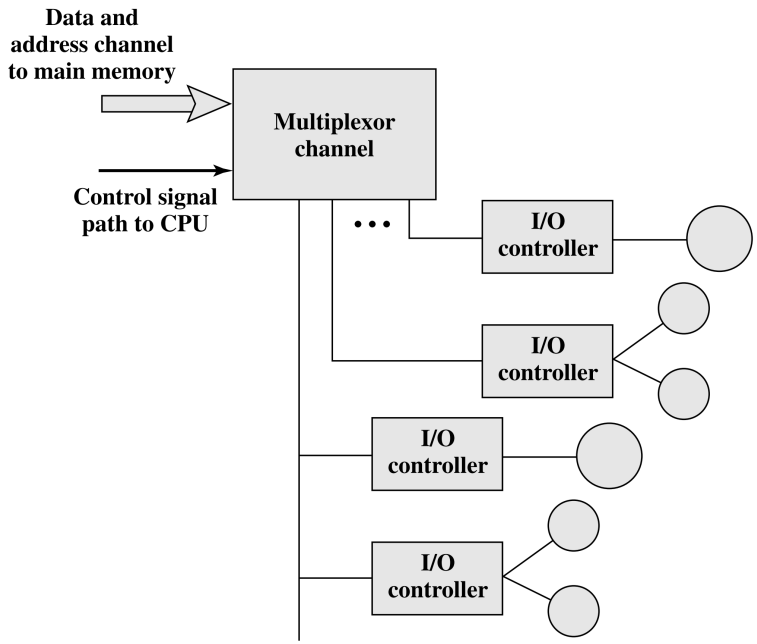
1. Canais de E/S são a evolução do DMA;
2. Executam instruções de E/S, o que dá completo controle sobre operações de E/S – ou seja, são processadores de propósito específico;
3. Por isso, a CPU não executa essas instruções, que estão em RAM para o Canal de E/S executar;
4. A CPU inicia a operação de E/S instruindo o Canal de E/S a executar um programa em RAM que especifica os dispositivos e áreas de memória para serem acessados, prioridades, ações para erros etc;

Processadores e Canais de E/S



Um canal do tipo **seletor** controla múltiplos dispositivos de E/S e, em um instante de tempo, é dedicado à transferência com um deles.

Processadores e Canais de E/S



Já um canal do tipo **multiplexador** múltiplos dispositivos de E/S ao mesmo tempo e faz o *interleaving* (intercalação). Para dispositivos lentos, intercalam-se bytes. Para dispositivos rápidos, blocos de dados.

$A_1 A_2 A_3 A_4 B_1 B_2 B_3 B_4 C_1 C_2 C_3 C_4 \rightarrow A_1 B_1 C_1 A_2 C_2 A_3 B_2 C_3 A_4$

Capítulo abordado: 7